IN THE CLAIMS:

1. (Currently Amended) A method, comprising the steps of:

generating a usage task from usage data;

constructing a pattern graph from the usage task;

constructing a model graph which represents a space of equivalents to the usage task represented by the pattern graph; and

extracting sub-graphs from the model graph, wherein each of the extracted sub-graphs is isomorphic to the pattern graph[[.]] ;

creating virtual tasks from each of the extracted subgraphs; and

performing a stress-test on an application program using the virtual tasks.

2. (Original) The method according to claim 1, further comprising the step of:

capturing the usage data from a user session.

3. (Original) The method according to claim 2, wherein the capturing step includes one of retrieving a log file of an application program and intercepting user calls to the application program.

4. (Original) The method according to claim 1, wherein the usage data includes API calls.

5. (Original) The method according to claim 1, wherein the generating task includes the sub-step of:

separating the usage data into generalizable entities and non-generalizable entities, wherein the pattern graph is constructed using only the generalizable entities.

6. (Original) The method according to claim 5, wherein the pattern graph construction step includes the sub-step of:

creating a fully qualified name for each of the generalizable entities, the fully qualified name including a local entity name and an associative qualifier.

7. (Original) The method according to claim 6, wherein the pattern graph construction step further includes the sub-steps of:

merging each fully qualified name into a node of the pattern graph; and

merging each of the associative qualifiers into an edge of the pattern graph.

8. (Original) The method according to claim 1, wherein the pattern graph is a directed acyclic graph.

9. (Original) The method according to claim 1, wherein the model graph is a directed acyclic graph.

10. (Original) The method according to claim 1, wherein the model graph retains all the dependencies of the pattern graph.

11. (Cancelled)

12. (Cancelled)

13. (Original) The method according to claim 1, wherein the model graph construction step includes the sub-step of:

determining an equivalent model graph node by matching node properties of a pattern graph node to node properties of a model graph node.

14. (Original) The method according to claim 1, wherein the extraction step includes the sub-step of:

performing one of a breadth-first search and a depth-first search of the model graph.

15. (Currently Amended) A system, comprising:

a usage task generation module configured to generate a usage task from usage data;

a pattern graph construction module configured to construct a pattern graph from ~~a~~ the usage task;

a model graph construction module configured to construct a model graph which represents a space of equivalents to the usage task represented by the pattern graph; ~~and~~

an extraction module configured to extract sub-graphs from the model graph, wherein each of the extracted sub-graphs is isomorphic to the pattern graph[[.]];

a virtual task creation module configured to create virtual tasks from each of the extracted subgraphs; and

a stress-test performance module configured to perform a stress-test on an application program using the virtual tasks.

16. (Original) The system according to claim 15, wherein the pattern graph includes nodes and edges.

17. (Original) The system according to claim 16, wherein each of the nodes includes a node label, the node label including a set of node properties for each of the nodes.

18. (Original) The system according to claim 17, wherein the node label is null.

19. (Original) The system according to claim 16, wherein each of the nodes includes a node value.

20. (Original) The system according to claim 16, wherein each of the edges is directed to reflect the ordering of the nodes.

21. (Original) The system according to claim 16, wherein the model graph also includes nodes and edges, wherein each of the nodes of the model graph is equivalent to at least one of the

nodes of the pattern graph and each of the edges of the model graph is equivalent to at least one of the edges of the pattern graph.

22. (Original) The system according to claim 16, wherein an equivalent model graph node is determined by matching node properties of one of the pattern graph nodes.

23. (Original) The method according to claim 15, wherein the pattern graph is a directed acyclic graph.

24. (Original) The method according to claim 15, wherein the model graph is a directed acyclic graph.

25. (Original) The method according to claim 15, wherein the model graph retains all the dependencies of the pattern graph.

26. (Original) A computer-readable storage medium storing a set of instructions, the set of instructions capable of being executed by a processor, the set of instructions performing the steps of:

generating a usage task from usage data;

constructing a pattern graph from the usage task;

constructing a model graph which represents a space of equivalents to the usage task represented by the pattern graph; and

extracting sub-graphs from the model graph, wherein each of the extracted sub-graphs is isomorphic to the pattern graph.

27. (New) The computer-readable storage medium of claim 26 wherein the set of instructions perform the further steps of:

creating virtual tasks from each of the extracted subgraphs; and

performing a stress-test on an application program using the virtual tasks.